Attorney's Docket No.: 13911-094001 / 2003P00732 US

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE:	CONTEXT OBJECTS FOR ACCESSING MESSAGE CONTENT
APPLICANT:	FRANK BEUNINGS, THEA HILLENBRAND AND UWE SCHLARB

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EV 399312375 US

March 12, 2004
Date of Deposit

CONTEXT OBJECTS FOR ACCESSING MESSAGE CONTENT

BACKGROUND

[0001] A collaborative computing environment employs some form of communication between computing systems, such as a messaging system for message exchange. In such systems, messages can take virtually any form, such as extensible markup language (XML) format, and can include any type of various content, such as a document, file, list, etc. Messaging systems typically use some form of routing scheme to guide messages from a sender to one or more intended recipients. Routing is conventionally directed by header information appended to the content of the message at or near the sender.

[0002] Increasingly, however, access to the content or "payload" of a message is often needed. For instance, business applications in a heterogeneous computing environment can communicate with each other through a message exchange infrastructure, such as provided by SAP of Walldorf, Germany. The exchange infrastructure and/or the applications may need to access the content of a request message from one application to another in order to fulfill their tasks. Several examples of these tasks include message routing and business process management. For example, a routing rule may be established, such as: "When Plant '01' is present in a message, then the receiver system is 'X'." In another example, a business process may require of an application, based on a message content: "When Sales Organization is '01' then wait for a confirmation."

[0003] Access to message content data is traditionally done via Xpath expressions, however such access is sometimes very cumbersome and typically very technical. An easier message content access technique is needed, and that can be employed across different interfaces.

SUMMARY

[0004] To improve access to message content in a message exchange system, a context object is defined for the message as an abstraction of the message content. The context object can be assigned to many different message communication interfaces, thereby allowing context objects to be reused across various interfaces. The context object provides access both to a message payload, and parts of a message other than the payload, such as technical data. Thus, access to message content for various tasks is improved, and applicability to a number of different message interfaces is enhanced.

[0005] The details of one or more embodiments are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] These and other aspects will now be described in detail with reference to the following drawings.

[0007] FIG. 1A is a block diagram of a computing environment suitably adapted for context objects.

[0008] FIG. 1B is a block diagram of an integration repository and integration directory of an exchange infrastructure.

[0009] FIG. 2 is an example message including context objects.

[0010] FIG. 3 is a runtime view of the integration repository in which context objects are stored.

[0011] Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0012] Systems and methods for improved access to message content in a message exchange system include a context object defined for message content. The context object is an abstraction of the message content, configurable by a user and preferably expressed as a string. These systems and methods further include the context object assigned to any interface for describing the message. Accordingly, context objects can be assigned to different interfaces, and can be reused across the different interfaces.

[0013] FIG. 1A is a block diagram of a computing environment 100 in which the use of context objects is suitably adapted. The computing environment 100 includes an exchange infrastructure (XI) 110. The XI 110 includes various adapters 209 configured to provide connectivity between the integration server 206 and proprietary applications 211, Web-based services 213, and third party applications 215. The XI 110 also can include a Web Application Server 210 providing Web-based applications. The Web Application Server 210 also includes a runtime engine 214 for messaging and business process control between Web-based applications such as Java applications 220, ABAP applications 222, and/or other software components.

[0014] A proxy generator 218 can generate interface proxies on demand for any application based on information stored on the integration repository 202. Web-based communication logic can be implemented based on the proxy that represents the interface description of the respective development platform, such as Java, ABAP, and .NET for the web-based applications 213. The proxies convert platform-specific data types into XML on the outbound side and vice versa on the inbound side and provide access to the component-specific local runtime engine 214.

[0015] With reference also to FIG. 1B, the integration repository 202 includes descriptions of business processes 232, context objects 234, mappings 236, and interfaces 238, all of which are defined according to one or more business scenarios 230. The business processes 232 are extensible compound Web services executed using a business process engine (not shown). The context objects 234 for message content can be defined in the integration repository 202, also according to the business scenarios, which describe and configure message-based interaction between applications. Each of the context objects 234 has a name and an associated namespace. For example, in keeping with the examples above, a context object 234 can be "*Plant*" or "*Sales Organization*" associated with the namespace "*http:/sap.com/xi/CRM.*"

[0016] Context objects 234 are predefined criteria to determine potential receivers of messages that must be distributed between software components and business partners during collaborative processing. Information about the context objects 234 are used in determining receiving application(s) prior to processing a complete message for distribution, and to be able to reuse certain criteria across different interfaces and provide a more abstract view on these interfaces. Mappings 236 define transformations between message interfaces 238, message types, or data types in the integration repository 202 that may be required.

[0017] The interface repository 202 also includes one or more interfaces 238 representing interface descriptions of all message interfaces of all software components in the environment. The interface repository 202 even provides interfaces that may potentially be used in a certain environment, whether or not actually used. Accordingly, the interfaces 238 can be implemented on any component using any technology. Message interfaces are made up of message types, which are in turn made up of data types. Interfaces 238 can be arranged according to any classification, such as inbound and outbound, or synchronous and asynchronous.

[0018] The integration directory 204 contains detailed collaboration knowledge that describes the configuration of each component as installed in the system. The integration directory 204 details information from the integration repository 202 that is specific to that configuration. The integration directory 204 includes descriptions of business scenarios 250, business processes 252, configured routing rules 254, and executable mappings 256. The integration directory 204 also includes descriptions of active Web services 258 and active business partners 260. The business scenarios 250 in the integration directory 204 represent the overall view of the interaction among interfaces and mappings 256 in the context of the actual configuration relevant for the specific implementation. The business processes 252 represents an executable description of all active business processes.

[0019] The routing rules 254 determine the receivers of a message on a business level. In one specific implementation, the content of a message is used as a routing rule 254. Other parameters may also be used. Relevant input parameters include the sender, the sender message type, the message to identify the receivers, and the receiver message type, and the content of the message as defined by the content object 234 associated with the message.

[0020] Mappings 256 in the integration directory 204 represent mappings required in the runtime system landscape, in contrast to the integration repository mappings 236 that contains all supported mappings. Web services 258 describe implemented interfaces of the current system landscape, as well as active Web services supported by business partners. Business partners 262 defines usual information for business partners of a company such as names, addresses, and URLs, but may also contain more detailed and sophisticated information.

[0021] The integration repository 202 and integration directory 204 use software component descriptions stored in the system landscape directory 203. The system landscape directory 203 includes design-time descriptions of components 240 for the integration repository 202 and a configuration-specific collaboration description of a system landscape 262 at runtime for the integration directory 204. The components 240 represent component descriptions that include information about application components, as well as information relating to their dependencies on each other. The system landscape 262 describes the runtime system landscape actively using the XI 110.

[0022] FIG. 2 illustrates one type of message in which access to application data is available in the message payload. The message includes a first context object 272 (representing "Sales Organization") and two or more second context objects 274 (representing "Plant"). Context objects can also be defined for messages in which access to application data is not available in the payload, or which only provide access to technical data.

[0023] FIG. 3 is a diagrammatical view of the repository 300 showing links from the business scenarios 302 to the business processes 304, and various outbound links.

The outbound links include references to local interfaces 306, references to context objects 308 and references to interface mappings 310. The local interfaces 306 can reference interfaces 312 and message types 314. The interface mappings 310 references message mappings 316.

[0024] The context objects 308 can be used to access payload information via name (e.g. 'Plant'). Data may not be written to context objects 308. Interface mappings 310 are addressed by a process within a transformation step.

[0025] In one exemplary situation, when the routing configuration is retrieved from the integration directory, the process name and the outbound interface are the sender information within the message. The message is submitted to the pipeline where the suitable routing relations are evaluated to determine target interface(s) and receiver(s). Context objects are used to distinguish from among different send steps on the same interface 306 or 308. When receivers are specified directly by process definition, the receiver (i.e. business system) may be entered directly or calculated by the receiver determination step. In this case, the pipeline must not calculate the receivers again upon the routing configuration.

[0026] If the routing configuration (directory) is used, context objects 308 are used to distinguish send steps that send messages of the same interface from different places in the process. In this manner, the send steps are each located in an exclusive branch of a switch element and need to be sent to different receivers. Without context information provided by context objects 308, it may not be possible to distinguish the send steps from each other because they are part of the same process and are being sent on the same interface. Routing can only use the process name and interface name. Context objects enable submission of additional context information via a *send* step to the routing. This information can be used in a routing condition and classify the routing relation true or false.

[0027] Although a few embodiments have been described in detail above, other modifications are possible. Other embodiments may be within the scope of the following claims.